Laboratorio de Programación 1 - 2024 S2

Tarea 1

Información general

Se sugiere leer con mucha atención todo el texto antes de comenzar a trabajar en esta tarea de laboratorio. Es muy importante que se respeten todos los requisitos solicitados en esta sección y las siguientes. Si surgen dudas, pedimos que las formulen en el foro del laboratorio en la página EVA del curso.

Individualidad

Esta tarea se deberá realizar en forma **individual**. Para todas las tareas de laboratorio rige el Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios (lectura obligatoria).

Calendario

Entrega: La entrega de la tarea puede realizarse hasta **las 16:00 del 17 de setiembre**. Los trabajos deberán ser entregados dentro de los plazos establecidos. **No** se aceptarán trabajos fuera de plazo. ***Para poder entregar, el estudiante debe haber realizado anteriormente el Cuestionario 2.***

Reentrega: Todos los estudiantes que realizaron la entrega pueden hacer modificaciones y realizar una segunda entrega (*reentrega*). La reentrega puede realizarse **hasta las 14:00 del 18 de setiembre.**

Forma de entrega

Se debe entregar un **único** archivo de nombre `tarea1.pas` que debe contener **únicamente** el código de los subprogramas pedidos y eventualmente el de subprogramas auxiliares que se necesiten para implementarlos. La entrega se realiza mediante una actividad en la plataforma EVA en la sección de laboratorio.

Archivos provistos y ejecución de pruebas

El archivo `definiciones.pas` contiene la constante `FINALIZADOR` definida por los docentes. El programa principal para realizar pruebas es provisto por los docentes en el archivo `principal.pas`. **No se debe modificar ninguno de estos archivos, dado que no serán entregados**. Para usar este programa se debe leer y seguir las instrucciones provistas en la sección cómo ejecutar los

casos de prueba de la Primera Tarea. Los casos de prueba que serán tomados en cuenta para la corrección de la tarea son los que se especifican en esa sección.

Introducción

La criptografía (del griego kryptós, «secreto», y graphé, «grafo» o «escritura», literalmente «escritura secreta») se ha definido, tradicionalmente, como el ámbito de la criptología que se ocupa de las técnicas de cifrado destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos incomprensibles a receptores no autorizados.

Un algoritmo de cifrado simple utilizado por Julio Cesar en el 58 AC para proteger la confidencialidad de sus mensajes con sus generales, y hoy nombrado en su honor, es un tipo de cifrado por sustitución. En este algoritmo, una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto. Julio Cesar en particular utilizaba un corrimiento de 3 lugares, es decir, la letra 'A' se sustituía por la letra 'D', la letra 'B' por la letra 'F' y así sucesivamente hasta llegar a la 'Z', que se sustituye por la 'C'. En la siguiente imagen se representa gráficamente el algoritmo de sustitución.

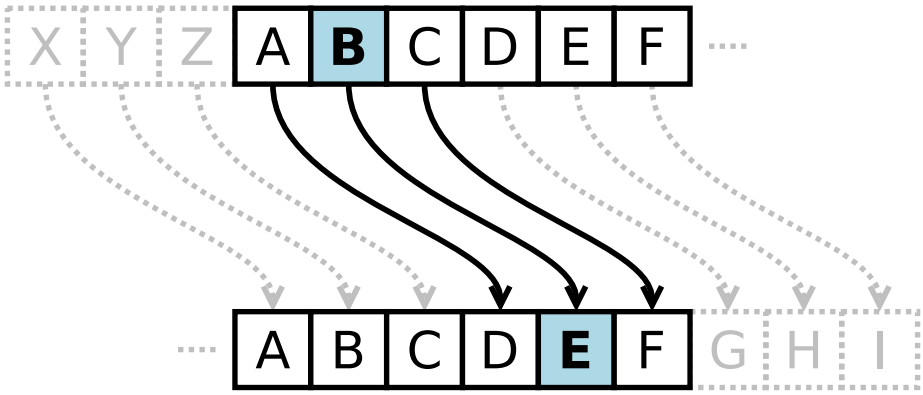

Figure 1: Ejemplo cifrado César

Versiones actuales del algoritmo utilizan una *clave* que indica el corrimiento, es decir el número de *desplazamiento* de las letras.

El estudiante debe implementar una función y un procedimiento, `sustituirLetra` y `procesarMensaje`, que serán invocados por el programa principal para sustituir una letra y leer una oración y cifrar el texto, respectivamente.

Subprogramas

Se deben implementar los siguientes subprogramas:

- Función `sustituirLetra`. Esta función recibe por parámetro la `letra` que se va a codificar y el `desplazamiento` que se aplicará. La función retorna el resultado de aplicar el algoritmo del Cesar. Solo se sustituirán los caracteres de la 'a' a la 'z' y de la 'A' a la 'Z'. Todo otro carácter no debe ser sustituido.

```
function sustituirLetra ( letra : char;
                          desplazamiento : integer)
                                          : char;
```

  Ejemplos:

  - `sustituirLetra('A',5)` retorna `'F'`
  - `sustituirLetra('C',-2)` retorna `'A'`
  - `sustituirLetra('Y',4)` retorna `'C'`
  - `sustituirLetra('a',5)` retorna `'f'`
  - `sustituirLetra('c',-2)` retorna `'a'`
  - `sustituirLetra('y',4)` retorna `'c'`
  - `sustituirLetra('!',5)` retorna `'!'`

- Procedimiento `procesarMensaje`. Recibe por parámetro la `clave` a utilizar en el algoritmo del Cesar y la `accion` (indicando con la letra 'C' para cifrar o 'D' para descifrar). El procedimiento deberá leer de la entrada estándar una oración que finaliza con el carácter `FINALIZADOR`. Para cada carácter leído se debe de invocar a la función `sustituirLetra` e ir desplegando en la salida estándar el mensaje procesado. En caso de que la `accion` sea el carácter 'C' implica que se cifrará el mensaje y el desplazamiento será la clave, en el caso que la `accion` sea 'D' se estará descifrando y el desplazamiento se calcula como el opuesto de la clave. Antes de desplegar la oración procesada, se deberá desplegar un mensaje indicando si se está cifrando o descifrando (ver ejemplos).

```
procedure procesarMensaje (clave: integer;
                           accion : char);
```

Ejemplos

A continuación se muestran ejemplos de ejecución del programa principal, asumiendo que el `FINALIZADOR` es `'.'`.

```
Ingrese la clave: 3
Ingrese la letra "C" para cifrar o "D" para descifrar: C
Ingrese el mensaje que desea procesar seguido de un punto ("."): hola.
El mensaje cifrado es: krod.
```

```
Ingrese la clave: 3
Ingrese la letra "C" para cifrar o "D" para descifrar: D
Ingrese el mensaje que desea procesar seguido de un punto ("."): krod.
El mensaje descifrado es: hola.
```

```
Ingrese la clave: 0
Ingrese la letra "C" para cifrar o "D" para descifrar: C
Ingrese el mensaje que desea procesar seguido de un punto ("."): Hola Mundo!.
El mensaje cifrado es: Hola Mundo!.
```

```
Ingrese la clave: 8
Ingrese la letra "C" para cifrar o "D" para descifrar: C
Ingrese el mensaje que desea procesar seguido de un punto ("."): Hola Mundo!.
El mensaje cifrado es: Pwti Ucvlw!.
```

Se pide

Escribir un archivo con todos los subprogramas solicitados. Los encabezados de los subprogramas **deben coincidir exactamente** con los que aparecen en esta letra. Si el estudiante realiza algún cambio se considerará que el subprograma no fue implementado. Si el estudiante lo desea, puede implementar subprogramas auxiliares adicionales (además de los subprogramas pedidos).

Para la corrección, las tareas se compilarán con una versión igual o posterior a **3.0.4 para Linux**. La compilación y la ejecución se realizarán en línea de comandos. El comando de compilación se invocará de la siguiente manera:

```
fpc -Co -Cr -Miso -gl principal.pas
```

`principal.pas` será el programa principal entregado por el equipo docente.

Debe compilar en la línea de comando. **NO** debe compilar usando ningún IDE.

No está permitido utilizar facilidades de Free Pascal que no forman parte del estándar y no se dan en el curso. Así por ejemplo, no se puede utilizar ninguna de las palabras siguientes: `uses`, `crlscr`, `gotoxy`, `crt`, `readkey`, `longint`, `string`, `break`, `exit`, etcétera.

En esta tarea, como en todos los problemas de este curso, se valorará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera, se hará énfasis en buenas prácticas de programación que lleven a un código legible, bien documentado y mantenible, tales como:

- indentación adecuada,
- utilización correcta y apropiada de las estructuras de control,
- código claro y legible,
- algoritmos razonablemente eficientes,

- utilización de comentarios que documenten y complementen el código,
- utilización de constantes simbólicas,
- nombres mnemotécnicos para variables, constantes, etcétera.

Para resolver la tarea se pueden utilizar todos los conocimientos vistos en el curso.